6 TOOL SUPPORT FOR TESTING

Peter Williams

INTRODUCTION

As seen in earlier chapters there are many tasks and activities that need to be performed during the testing process. In addition, other tasks need to be performed to support the testing process.

In order to assist in making the testing process easier to perform and manage, many different types of test tools have been developed and used for a wide variety of testing tasks. Some of them have been developed in-house by an organisation's own software development or testing department. Others have been developed by software houses (also known as test-tool vendors) to sell to organisations that perform testing. Even within the same type of tool, some will be home-grown and others will be developed by test-tool vendors.

This chapter discusses the potential benefits and pitfalls associated with test tools in general. It then describes the most commonly used types of test tools and concludes with a process for introducing a tool into a test organisation.

Learning objectives

The learning objectives for this chapter are listed below. You can confirm that you have achieved these by using the self-assessment questions at the start of the chapter, the 'Check of understanding' boxes distributed throughout the text, and the example examination questions provided at the end of the chapter. The chapter summary will remind you of the key ideas.

The sections are allocated a K number to represent the level of understanding required for that section; where an individual section has a lower K number than the section as a whole this is indicated for that topic; for an explanation of the K numbers see the Introduction.

Types of test tool (K2)

- Classify different types of test tools according to their purpose and to the activities of the fundamental test process and the software life cycle.
- Explain the term test tool and the purpose of tool support for testing.

Effective use of tools: potential benefits and risks (K2)

- Summarise the potential benefits and risks of test automation and tool support for testing.
- Remember special considerations for test execution tools, static analysis and test management tools. (K1)

Introducing a tool into an organisation (K1)

- State the main principles of introducing a tool into an organisation.
- State the goals of a proof-of-concept for tool evaluation and a piloting phase for tool implementation.
- Recognise that factors other than simply acquiring a tool are required for good tool support.

Self-assessment questions

The following questions have been designed to enable you to check your current level of understanding for the topics in this chapter. The answers are at the end of the chapter.

Question SA1 (K2)

Which of the following pairs of test tools are **likely** to be **most useful** during the test analysis and design stage of the fundamental test process?

(i) Test execution tool
(ii) Test data preparation tool
(iii) Test management tool
(iv) Requirements management tool

a. (i) and (ii)
b. (i) and (iv)
c. (ii) and (iii)
d. (iii) and (iv)

Question SA2 (K2)

Which of the following is **most likely** to cause failure in the implementation of a test tool?

a. Underestimating the demand for a tool.
b. The purchase price of the tool.
c. No agreed requirements for the tool.
d. The cost of resources to implement and maintain the tool.

Question SA3 (K2)

What benefits do static analysis tools have over test execution tools?

a. Static analysis tools find defects earlier in the life cycle.
b. Static analysis tools can be used before code is written.
c. Static analysis tools test that the delivered code meets business requirements.
d. Static analysis tools are particularly effective for regression testing.

WHAT IS A TEST TOOL?

Definition of a test tool

The ISTQB Glossary of Testing Terms defines a test tool as:

> A software product that supports one or more test activities, such as planning and control, specification, building initial files and data, test execution and test analysis.

Therefore a test tool can be thought of as a piece of software that is used to make the testing process more effective or efficient. In other words, anything that makes testing easier, quicker, more accurate, etc.

This book will focus on those test tools that are listed in the syllabus. These are, in general, the test tools that are most commonly used in the testing process and designed primarily for the testing process.

Benefits and risks of using any type of tool

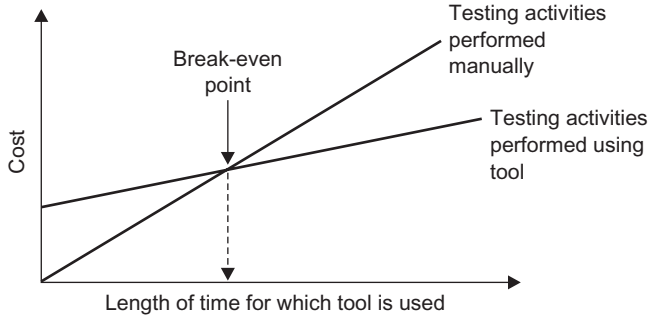
Let us consider the building of a new hotel and examine the similarities with the introduction and use of test tools. Test tools need to be thought of as long-term investments that need maintenance to provide long-term benefits. Similarly, building a hotel requires a lot of upfront planning, effort and investment. Even when the hotel is ready for use, there is still a continual long-term requirement for the provision of services such as catering, cleaning, building maintenance, provision of staff to provide ad hoc services to customers, etc. The long-term benefit is that this upfront investment and ongoing maintenance and support can provide substantial income in return.

In addition, there are risks that over a period of time, the location of the hotel will become less attractive, resulting in lower demand, lower usage and a maintenance cost that is greater than the income received. Therefore the initial investment is wasted because the ongoing need/requirement did not exist.

The graph in Figure 6.1 demonstrates a typical payback model for implementing a test execution tool. The same principle applies to the majority of test tools. Note that there is an ongoing maintenance cost of using the tool, but that this ongoing maintenance cost needs to be less than the cost of performing testing activities without the tool if the investment is to be worthwhile.

The same advantages and disadvantages apply to the use of most types of test tool. However, there are exceptions to this generalisation (and to the same generalisation made in the ISTQB syllabus). Some tools, such as comparators, can be used virtually straight out of the box. A comparator can check whether one large test file is the same as another. If it is different it can identify and report upon the differences. This would be very difficult and time-consuming to do manually. In addition, incident management tools are fairly intuitive and easy for both experienced and novice testers to use. They are also likely to provide a 'quick win'.

Figure 6.1 Test tool payback model



Other tools can be built by developers in-house as the need arises. For instance, test harnesses, test oracles or test data preparation tools may be relatively easy to produce for developers with a good understanding of the tool requirements and the systems and databases in the test environment.

Benefits

The main benefit of using test tools is similar to the main benefit of automating any process. That is, the amount of time and effort spent performing routine, mundane, repetitive tasks is greatly reduced. For example, consider the time and cost of making consumer goods by hand or in a factory.

This time saved can be used to reduce the costs of testing or it can be used to allow testers to spend more time on the more intellectual tasks of test planning, analysis and design. In turn, this can enable more focused and appropriate testing to be done – rather than having many testers working long hours, running hundreds of tests.

Related to this is the fact that the automation of any process usually results in more predictable and consistent results. Similarly, the use of test tools provides more predictable and consistent results as human failings such as manual-keying errors, misunderstandings, incorrect assumptions, forgetfulness, etc., are eliminated. It also means that any reports or findings tend to be objective rather than subjective. For instance, humans often assume that something that seems reasonable is correct, when in fact it may not be what the system is supposed to do.

The widespread use of databases to hold the data input, processed or captured by the test tool, means that it is generally much easier and quicker to obtain and present accurate test management information, such as test progress, incidents found/fixed, etc. (see Chapter 5).

Risks

Most of the risks associated with the use of test tools are concerned with over-optimistic expectations of what the tool can do and a lack of appreciation of the effort required to implement and obtain the benefits that the tool can bring.

For example, consider the production environments of most organisations considering using test tools. They are unlikely to have been designed and built with test tools in mind. Therefore, assuming that you want a test environment to be a copy of production (or at least as close to it as possible), you will also have a test environment that is not designed and built with test tools in mind.

Consider the test environments used by vendors to demonstrate their test tools. If you were the vendor would you design the environment to enable you to demonstrate the tool at its best or to demonstrate the shortcomings it may encounter in a typical test environment?

Therefore, unless detailed analysis and evaluation is done, it is likely that test tools will end up as something that seemed a good idea at the time but have been largely a waste of time and money. A process for avoiding such problems when introducing a tool into an organisation is described later in this chapter.

After a test tool has been implemented and measurable benefits are being achieved, it is important to put in sufficient effort to maintain the tool, the processes surrounding it and the test environment in which it is used. Otherwise there is a risk that the benefits being obtained will decrease and the tool will become redundant. Additionally, opportunities for improving the way in which the tool is used could also be missed.

For example, the acquisition of various test tools from multiple vendors will require interfaces to be built or configured to import and export data between tools. Otherwise much time may be spent manually cutting and pasting data from one tool to another. If this is not done, then data inconsistencies and version control problems are likely to arise. Similar problems may arise when testing with third-party suppliers or as a result of mergers and acquisitions.

Maintenance effort will also be required to upgrade and re-configure tools so that they remain compliant with new platforms or operating systems.

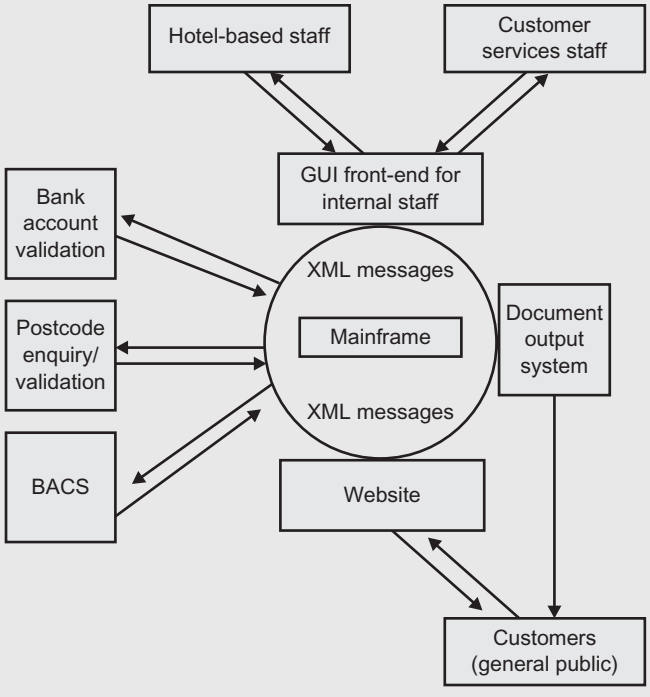
EXAMPLE – HOTEL CHAIN SCENARIO

An example of a hotel chain with several UK-based hotels will be used throughout this chapter. The systems that comprise the organisation's system architecture are shown in Figure 6.2.

The general public can book rooms at any of the chain's hotels by:

- Contacting staff in the hotel, who then use a GUI front-end to make the booking.
- Telephoning customer services who then use a GUI front-end to make the booking.
- Using the company's website to make the booking online.

Figure 6.2 Hotel system architecture



In all cases, communication with the mainframe computer is done via a middleware layer of XML messages.

There is a document production system that produces paper and electronic versions of customer correspondence such as booking confirmations, bills, invoices, etc.

Direct debit and credit card payments are made via BACS. Files are transmitted and confirmation and error messages are received back.

Validation of bank account details is performed by sending XML messages to and from a third-party system.

Validation and enquiry of address and postcode is also performed by sending XML messages to and from a third-party system.

A new release of the system is planned for six months' time. This will include:

- Code changes to improve performance in the XML middleware layer and on the mainframe. Mainframe changes will be performed by an outsourced development team in India.
- Various changes to website screens to improve usability.

- The introduction of a new third-party calendar object from which dates can be selected.
- The ability for customers to pay by cheque.
- The automatic production of cheques for refunds, cancellations, etc.
- An amended customer bill plus two other amended documents.
- Two new output documents.
- Fixes to various existing low- and medium-severity defects.

CHECK OF UNDERSTANDING

(1) Would you expect a quick return on your investment in test tools? Why?

(2) Describe three potential benefits of using test tools.

(3) Describe two risks of using test tools.

TEST TOOLS

Types of tool

There are several ways in which test tools can be classified. They can be classified according to:

- their purpose;
- the activities within the fundamental test process and the software life cycle with which they are primarily associated;
- the type of testing that they support;
- the source of tool (shareware, open source, free or commercial);
- the technology used;
- who uses them.

In this book, test tools will be classified according to the type of activity they support (as in the ISTQB Foundation Level Syllabus).

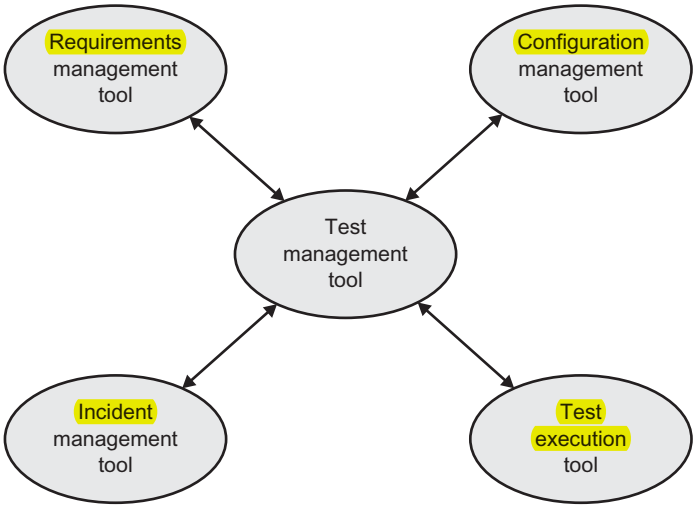
Tool support for management of testing and tests

Test management tools

Test management tools provide support for various activities and tasks throughout the development life cycle. Some of these activities are supported by the provision of interfaces to other more specialist tools (for instance, test execution tools). Other test management tools provide fully integrated modules that provide some or all of the services/functions provided by more specialist tools (such as incident management or requirements management tools).

The diagram in Figure 6.3 shows how a test management tool is the hub or centre of a set of integrated test tools.

Figure 6.3 An integrated set of tools



Test management tools provide an architecture for creating, storing and editing test procedures. These may be linked or traced to requirements, test conditions and risks. Such test procedures can then be prioritised or grouped together and scheduled so that they are run in the most effective and efficient order. Some test management tools allow dependencies to be recorded so that tests that will fail owing to a known defect can be highlighted and left unexecuted. This allows testers to be redirected to run the highest priority tests available rather than waste their time and the test data they have prepared on tests that are certain to fail.

Tests can be recorded as passed or failed and usually a test management tool provides an interface to an incident management tool so that an incident can be raised if the actual and expected results differ.

Test management tools can provide management information and reports on test procedures passed or failed. The amount of integration with other specialist tools is significant here. For instance, integration with requirements management tools allows reports to be produced on test progress against one or more requirements. Integration with incident management tools allows reports also to include analysis of defects against requirements.

Test management tools generally hold data in a database. This allows a large amount of reports and metrics to be produced. The metrics produced can be used as inputs to:

- Test and project management to control the current project.

- Estimates for future projects.
- Identifying weaknesses or inefficiencies in the development or test process that can be subsequently investigated with the aim of improving them.

Test management information reports should be designed to meet the needs of project managers and other key users. It may be necessary to export data to a spreadsheet in order for it to be manipulated into the format required.

A test management tool can enable reuse of existing testware in future test projects.

USE IN HOTEL CHAIN SCENARIO

In the scenario, a test management tool can be used to write down and store requirements for new functionality and subsequently to hold the test conditions necessary to test these requirements.

It can also be used to record whether tests have passed or failed and to produce test management information on progress to date.

Additionally, requirements and test conditions from previous developments will already exist in the test management tool. These can be used as the basis for the regression testing required. Indeed a regression pack may already exist. Clearly the regression pack would have to be reviewed and amended as necessary to make it relevant to this release. However, the benefit is that much of the previous work could be reused, which, in turn, means that much less effort will be involved to create a regression pack.

Incident management tools

Incident management tools (also known as defect management tools) are one of the most widely used types of test tool. At a basic level incident management tools are used to perform two critical activities: creation of an incident report; and maintenance of details about the incident as it progresses through the incident life cycle.

The level of detail to be captured about the incident can be varied depending upon the characteristics of the tool itself and the way in which the incident management tool is configured and used by the test organisation.

For example, the incident management tool could be configured so that lots of mandatory information is required in order to comply with industry or generic standards such as IEEE 1044. In addition, workflow rules may also be applied to ensure that the agreed incident life cycle is strictly applied, with incidents only able to be assigned to particular teams or users. Alternatively, the tool could be configured to require very limited mandatory information, with most fields being free format.

Incident management tools also use a database to store and manage details of incidents. This allows the incident to be categorised according to the values stored in appropriate fields. Such values will change during the incident life cycle as the incident is analysed, debugged, fixed and re-tested. It is often possible to view the history of changes made to the incident.

The database structure also enables incidents to be searched and analysed (using either filters or more complex SQL-type queries). This provides the basis for management information about incidents. Note that as the values held against each incident change, the management information will also change. Therefore users need to be aware of the danger of using outdated reports.

This data can also be used in conjunction with data held in test management tools when planning and estimating for future projects. It can also be analysed to provide input to test process improvement projects.

Fields in the database structure normally include:

- Priority (e.g. high, medium, low).
- Severity (e.g. high, medium, low).
- Assignee (the person to whom the incident is currently assigned, e.g. a developer for debugging, a tester to perform retesting).
- Status in the incident life cycle (e.g. New, Open, Fixed, Reopen, Closed).

This would allow management information to be produced from the incident management database about the number of high-priority incidents with a status of Open or Reopen that are assigned to, say, Peter Morgan, compared with the number assigned to Brian Hambling.

Some test management tools include fully integrated incident management tools as part of their core product, whilst other incident management tools can be integrated with test management, requirements management and/or test execution tools. Such integration enables incidents to be input and traced back to test cases and requirements.

USE IN HOTEL CHAIN SCENARIO

An incident management tool can be used to raise new defects and process them through the defect life cycle until resolved. It can also be used to check whether defects (or similar defects) have been raised before, especially for defects raised during regression testing.

An incident management tool could also be used to prioritise defects so that developers fix the most important ones first. It could also highlight clusters of defects. This may suggest that more detailed testing needs to be done on the areas of functionality where most defects are being found as it is probable that further defects will be found as well.

Requirements management tools

Requirements management tools are used by business analysts to record, manage and prioritise the requirements of a system. They can also be used to manage changes to requirements – something that can be a significant problem for testers as test cases are designed and executed to establish whether the delivered system meets its requirements. Therefore if requirements change after tests have been written then test cases may also need to change. There is also a potential problem of changes not being communicated to all interested parties, thus testers could be using an old set of requirements whilst new ones are being issued to developers.

The use of a traceability function within a requirements tool (and/or integrated with a test management tool) enables links and references to be made between requirements, functions, test conditions and other testware items. This means that as requirements change, it is easy to identify which other items may need to change.

Some requirements management tools can be integrated with test management tools, whilst some test management tools enable requirements to be input and related to test cases.

Requirements management tools also enable requirements coverage metrics to be calculated easily as traceability enables test cases to be mapped to requirements.

As can be seen, traceability can create a lot of maintenance work, but it does highlight those areas that are undergoing change.

USE IN HOTEL CHAIN SCENARIO

A change is required to three documents sent to customers. The requirements are documented in the requirements management tool. Testers obtain the requirements from the tool and begin to devise test conditions and test cases. A subsequent change control means that further changes are made to the requirements. The testers should be made aware of the changes so that they can provide input to the impact analysis. However, traceability within a requirements management tool will also highlight the test conditions affected by the changed requirement. The testers can review the change in requirements and then consider what changes need to be made to the test conditions and test cases.

Configuration management tools

Configuration management tools are designed primarily for managing: the versions of different software (and hardware) components that comprise a complete build of the system; and various complete builds of systems that exist for various software platforms over a period of time.

A build is a development activity where a complete system is compiled and linked (typically daily) so that a consistent system is available at any time including all the latest changes.

USE IN HOTEL CHAIN SCENARIO

Within the hotel booking system, there will be many versions of subsystems due to the date at which the version was included in a build, or the operating system on which the version works, etc. Each version of a subsystem will have a unique version number and each version of a subsystem will comprise many different components (e.g. program files, data files, DLLs, etc.).

The configuration management tool maps the version number of each subsystem to the build (or release) number of the integrated system. As shown in Table 6.1, Build A (UNIX) and Build B (Microsoft Windows 2003) might use the same version (v1.02) of the Payments Out subsystem, but Release C might use version v1.04.

Table 6.1 Configuration traceability

Build for integrated system	Version of Payments Out system	Cheque test procedure ID	Check BACS file test procedure ID
Build A	v1.02	TP123a	TP201
Build B	v1.02	TP123b	TP201
Build C	v1.04	TP123b	TP201

The same principle applies to testware with a different version number for a test procedure being used, depending upon the version number of the build. For instance, test procedure TP123a might be used for Build A and TP123b might be used for Build B – even though both have the same purpose and even expected results. However, another test procedure, TP201, may be applicable to all builds.

The amount of benefit to be obtained from using configuration management tools is largely dependent upon:

- the complexity of the system architecture;
- the number and frequency of builds of the integrated system;
- how much choice (options) is available to customers (whether internal or external).

For example, a software house selling different versions of a product to many customers who run on a variety of operating systems is likely to find configuration management tools more useful than an internal development department working on a single operating system for a single customer.

The use of configuration management tools allows traceability between testware and builds of an integrated system and versions of subsystems and modules. Traceability is useful for:

- identifying the correct version of test procedures to be used;
- determining which test procedures and other testware can be reused or need to be updated/maintained;
- assisting the debugging process so that a failure found when running a test procedure can be traced back to the appropriate version of a subsystem.

CHECK OF UNDERSTANDING

(1) What is traceability?

(2) Which tool is likely to be most closely integrated with a requirements management tool?

(3) Which tool would you use to identify the version of the software component being tested?

Tool support for static testing

Review tools

Review tools (also known as review process support tools) provide a framework for reviews or inspections. This can include:

- Maintaining information about the review process, such as rules and checklists.
- The ability to record, communicate and retain review comments and defects.
- The ability to amend and reissue the deliverable under review whilst retaining a history or log of the changes made.
- Traceability functions to enable changes to deliverables under review to highlight other deliverables that may be affected by the change.
- The use of web technology to provide access from any geographical location to this information.

Review tools can interface with configuration management tools to control the version numbers of a document under review.

If reviews and inspections are already performed effectively then a review tool can be implemented fairly quickly and relatively cheaply. However, if such a tool is used as a means for imposing the use of reviews then the training and

implementation costs will be fairly high (as would be the case for implementing a review process without such tools). These tools support the review process, but management buy-in to reviews is necessary if benefits from them are to be obtained in the long run.

Review tools tend to be more beneficial for peer (or technical) reviews and inspections rather than walkthroughs and informal reviews.

USE IN HOTEL CHAIN SCENARIO

The hotel company could use a review tool to perform a review of a system specification written in the UK, so that offshore developers can be involved in the review process. In turn, the review of program code, written offshore, could also be performed using such a tool. This means that both the UK and offshore staff could be involved in both reviews, with no excuses for the right people not being available to attend.

Static analysis tools

Static analysis tools analyse code before it is executed in order to identify defects as early as possible. Therefore they are used mainly by developers prior to unit testing. A static analysis tool generates lots of error and warning messages about the code. Training may be required in order to interpret these messages and it may also be necessary to configure the tool to filter out particular types of warning messages that are not relevant. The use of static analysis tools on existing or amended code is likely to result in lots of messages concerning programming standards. A way of dealing with this situation should be considered during the selection and implementation process. For instance, it may be agreed that small changes to existing code should not use the static analysis tool whereas medium to large changes to existing code should use the static analysis tool. A rewrite should be considered if the existing code is significantly non-compliant.

Static analysis tools can find defects that are hard to find during dynamic testing. They can also be used to enforce programming standards (including secure coding), improve the understanding of the code and to calculate complexity and other metrics (see Chapter 3).

Some static analysis tools are integrated with dynamic analysis tools and coverage measurement tools. They are usually language specific, so to test code written in C++ you would need to have a version of a static analysis tool that was specific to C++.

Other static analysis tools come as part of programming languages or only work with particular development platforms. Note that debugging tools and compilers provide some basic functions of a static analysis tool, but they are generally not considered to be test tools and are excluded from the ISTQB syllabus.

The types of defect that can be found using a static analysis tool can include:

- Syntax errors (e.g. spelling or missing punctuation).
- Variance from programming standards (e.g. too difficult to maintain).
- Invalid code structures (missing ENDIF statements).
- The structure of the code means that some modules or sections of code may not be executed. Such unreachable code or invalid code dependencies may point to errors in code structure.
- Portability (e.g. code compiles on Windows but not on UNIX).
- Security vulnerabilities.
- Inconsistent interfaces between components (e.g. XML messages produced by component A are not of the correct format to be read by component B).
- References to variables that have a null value or variables declared but never used.

USE IN HOTEL CHAIN SCENARIO

Static analysis tools may be considered worthwhile for code being developed by offshore development teams who are not familiar with in-house coding standards. Such tools may also be considered beneficial for high-risk functions such as BACS and other external interfaces.

Modelling tools

Modelling tools are used primarily by developers during the analysis and design stages of the development life cycle. The reason modelling tools are included here is because they are very cost-effective at finding defects early in the development life cycle.

Their benefits are similar to those obtained from the use of reviews and inspections, in that modelling tools allow omissions and inconsistencies to be identified and fixed early so that detailed design and programming can begin from a consistent and robust model. This in turn prevents fault multiplication that can occur if developers build from the wrong model.

For instance, a visual modelling tool using UML can be used by designers to build a model of the software specification. The tool can map business processes to the system architecture model, which, in turn, enables programmers and testers to have a better and common understanding of what programs should do and what testing is required.

Similarly, the use of database, state or object models can help to identify what testing is required and can assist in checking whether tests cover all necessary transactions. Integration with test design tools may also enable modelling tools to support the generation of test cases.

USE IN HOTEL CHAIN SCENARIO

The modelling tool could help to identify missing scenarios from letter templates or the need to update letters with new paragraphs. Again, the benefits of a clearly defined, consistent model of the software will assist offshore companies to develop software that meets the requirements of the customer.

The use of modelling tools is particularly useful in complex system architectures such as in this scenario.

CHECK OF UNDERSTANDING

(1) Which of the tools used for static testing is/are most likely to be used by developers rather than testers?

(2) In which part of the fundamental test process are static analysis tools likely to be most useful?

(3) What is a significant benefit of using modelling tools from a testing perspective?

Tool support for test specification

Test design tools

Test design tools are used to support the generation and creation of test cases. In order for the tool to generate test cases, a test basis needs to be input and maintained. Therefore many test design tools are integrated with other tools that already contain details of the test basis such as:

- modelling tools;
- requirements management tools;
- static analysis tools;
- test management tools.

The level of automation can vary and depends upon the characteristics of the tool itself and the way in which the test basis is recorded in the tool. For example, some tools allow specifications or requirements to be specified in a formal language. This can allow test cases with inputs and expected results to be generated. Other test design tools allow a GUI model of the test basis to be created and then allow tests to be generated from this model.

Some tools (sometimes known as test frames) merely generate a partly filled template from the requirement specification held in narrative form. The tester will then need to add to the template and copy and edit as necessary to create the test cases required.

Tests designed from database, object or state models held in modelling tools can be used to verify that the model has been built correctly and can be used to derive some test cases. Tests derived can be very thorough and give high levels of coverage in certain areas.

Some static analysis tools integrate with tools that generate test cases from an analysis of the code. These can include test input values and expected results.

A test oracle is a type of test design tool that automatically generates expected results. However, these are rarely available as they perform the same function as the software under test. Test oracles tend to be most useful for:

- Replacement systems
- Migrations
- Regression testing

USE IN HOTEL CHAIN SCENARIO

A test oracle could be built using a spreadsheet to support the testing of customers' bills. The tester can then input details for calculating bills such as the total bill based on various transaction types, refunds, VAT, etc. The spreadsheet could then calculate the total bill amount and this should match the bill calculated by the system under test.

However, test design tools should be only part of the approach to test design. They need to be supplemented by other test cases designed with the use of other techniques and the application of risk.

Test design tools could be used by the test organisation in the scenario but the overhead to input the necessary data from the test basis may be too great to give any real overall benefit. However, if the test design tool can import requirements or other aspects of the test basis easily then it may become worthwhile.

Test design tools tend to be more useful for safety-critical and other high-risk software where coverage levels are higher and industry, defence or government standards need to be adhered to. Commercial software applications, like the hotel system, do not usually require such high standards and therefore test design tools are of less benefit in such cases.

Test data preparation tools

Test data preparation tools are used by testers and developers to manipulate data so that the environment is in the appropriate state for the test to be run. This can involve making changes to the field values in databases, data files, etc., and populating files with a spread of data, (including depersonalised dates of birth, names and addresses, etc. to support data anonymity).

USE IN HOTEL CHAIN SCENARIO

A set of test data may be created by taking, say, 5 per cent of all records from the live system and scrambling personal details so that data is protected and to ensure that customer letters being tested are not wrongly sent to real customers. Data could be taken from the mainframe system, but it is also very important to retain integrity of data between different systems. Data that is held in other databases would need to remain consistent with records on the mainframe.

The knowledge of the database structure and which fields need to be depersonalised is likely to lie with the development team – so it is important to consider whether to buy a tool or build it within the organisation.

CHECK OF UNDERSTANDING

(1) What is the main difference and similarity between a test frame and a test oracle?

(2) What types of inputs can a test design tool use to generate test cases?

Tool support for test execution and logging

Test comparators

Test comparators compare the contents of files, databases, XML messages, objects and other electronic data formats. This allows expected results and actual results to be compared. They can also highlight differences and thus provide assistance to developers when localising and debugging code.

They often have functions that allow specified sections of the file, screen or object to be ignored or masked out. This means that a date or time stamp on a screen or field can be masked out as it is expected to be different when a comparison is performed.

Table 6.2 shows an extract from the transaction table in the hotel chain database for data created on 20/10/2006.

Table 6.2 Hotel system extract (20/10/2006)

Transaction ID	Trans_Date	Amount _exc_VAT	VAT	Customer ID
12345	20/10/2006	359.66	62.94	AG0012
12346	20/10/2006	2312.01	404.60	HJ0007

A regression test was run on 5/11/2006. Table 6.3 shows an extract from the transaction table for this data.

Table 6.3 Hotel system extract (5/11/2006)

Transaction ID	Trans_Date	Amount _exc_VAT	VAT	Customer ID
12369	5/11/2006	359.66	62.94	AG0012
12370	5/11/2006	2312.01	404.60	HJ0007

The Transaction ID and Trans_Date fields contain different values. But we know why this is the case and we would expect them to be different. Therefore we can mask out these values. Note that some automated test comparators use test oracles whilst others provide functions to add on values to take into account known differences (e.g. 15 days later) so that the actual results and expected results can be compared.

Comparators are particularly useful for regression testing since the contents of output or interface files should usually be the same. This is probably the test tool that provides the single greatest benefit. For instance, manually comparing the contents of a database query containing thousands of rows is time-consuming, error prone and demotivating. The same task can be performed accurately and in a fraction of the time using a comparator. Comparators are usually included in test execution tools.

Test execution tools

Test execution tools allow test scripts to be run automatically (or at least semi-automatically). A test script (written in a programming language or scripting language) is used to navigate through the system under test and to compare predefined expected outcomes with actual outcomes. The results of the test run are written to a test log. Test scripts can then be amended and reused to run other or additional scenarios through the same system. Some tools offer GUI-based utilities that enable amendments to be made to scripts more easily than by changing code. These utilities may include:

- configuring the script to identify particular GUI objects;
- customising the script to allow it to take specified actions when encountering particular GUI objects or messages;
- parameterising the script to read data from various sources.

Record (or capture playback) tools: Record (or capture playback) tools can be used to record a test script and then play it back exactly as it was executed. However, a test script usually fails when played back owing to unexpected results

or unrecognised objects. This may sound surprising but consider entering a new customer record onto a system:

- When the script was recorded, the customer record did not exist. When the script is played back the system correctly recognises that this customer record already exists and produces a different response, thus causing the test script to fail.
- When a test script is played back and actual and expected results are compared a date or time may be displayed. The comparison facility will spot this difference and report a failure.
- Other problems include the inability of test execution tools to recognise some types of GUI control or object. This might be able to be resolved by coding or reconfiguring the object characteristics (but this can be quite complicated and should be left to experts in the tool).

Also note that expected results are not necessarily captured when recording user actions and therefore may not be compared during playback.

The recording of tests can be useful during exploratory testing for reproducing a defect or for documenting how to execute a test. In addition, such tools can be used to capture user actions so that the navigation through a system can be recorded. In both cases, the script can then be made more robust by a technical expert so that it handles valid system behaviours depending upon the inputs and the state of the system under test.

Data-driven testing: Robust test scripts that deal with various inputs can be converted into data-driven tests. This is where hard-coded inputs in the test script are replaced with variables that point to data in a data-table. Data-tables are usually spreadsheets with one test case per row, with each row containing test inputs and expected results. The test script reads the appropriate data value from the data-table and inserts it at the appropriate point in the script (e.g. the value in the Customer Name column is inserted into the Customer Name field on the input screen).

Keyword-driven testing: A further enhancement to data-driven testing is the use of keyword-driven (or action word) testing. Keywords are included as extra columns in the data-table. The script reads the keyword and takes the appropriate actions and subsequent path through the system under test. Conditional programming constructs such as IF ELSE statements or SELECT CASE statements are required in the test script for keyword-driven testing.

Technical skills: Programming skills and programming standards are required to use the tool effectively. It may be that these can be provided by a small team of technical experts within the test organisation or from an external company. In data-driven and particularly keyword-driven approaches, the bulk of the work can be done by manual testers, with no knowledge of the scripting language, defining their test cases and test data and then running their tests and raising defects as required. However, this relies on robust and well-written test scripts that are easy to maintain. This takes much time and effort before any sort of payback is achieved.

Maintenance: It is essential that time (and subsequent budget) is allowed for test scripts to be maintained. Any change to a system can mean that the test scripts need to be updated. Therefore, the introduction of a new type of object or control could result in a mismatch being found between the previous object type and the new one. The relevant level of technical skills and knowledge is also required to do this.

Effective and efficient use: The efficiency and effectiveness benefits that come from the use of a test execution tool take a long time to come to fruition. First, the selection and implementation process needs to be planned and conducted effectively (a generic process for selecting and implementing any type of test tool is detailed later in this chapter). However, there are certain issues that are particularly relevant to test execution tools and these are described below.

The long-term benefits of test execution tools include:

- Cost savings as a result of the time saved by running automated tests rather than manual tests.
- Accuracy benefits from avoiding manual errors in execution and comparison.
- The ability and flexibility to use skilled testers on more useful and interesting tasks (than running repetitive manual tests).
- The speed with which the results of the regression pack can be obtained.

Note that benefits come primarily from running the same or very similar tests a number of times on a stable platform. Therefore they are generally most useful for regression testing.

USE IN HOTEL CHAIN SCENARIO

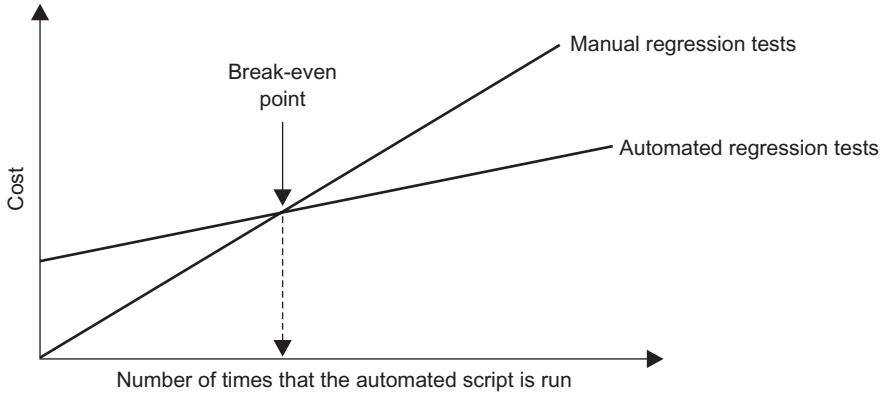
Let us assume that a test execution tool is already used for regression testing. Existing automated test scripts could be analysed to identify which ones can be reused and to identify gaps in the coverage for the new enhancement. These gaps could be filled by running cases manually or by writing new automated test scripts. Rather than starting from scratch, it may be possible to produce additional automated scripts by reusing some code or modules already used by existing scripts, or by using parameterisation and customisation utilities. In this enhancement, the automated scripts used to test the unchanged documents could be run without having to be amended.

The automated scripts to produce the amended documents would need to be analysed and updated as required. The navigation part of the script would be largely unchanged but the comparison between actual and expected results would probably be performed manually the first time round. Once the test has passed manually, the comparison could be added to the script for reuse in the future.

Automated scripts for new documents could be added to the regression pack after this release is complete.

The graph in Figure 6.4 shows how the benefits of using test execution tools take some time to pay back. Note how in the early stages the cost of using automated regression testing is greater than the cost of manual regression testing. This is due to the initial investment, implementation, training, initial development of automated scripts, etc. However, the cost each additional time the test is run is less for automated regression testing than it is for manual regression testing. Therefore the lines on the graph converge and at a point in time (known as the break-even point) the lines cross. This is the point at which the total cost to date for automated testing is less than the total cost to date for manual regression testing.

Figure 6.4 Test execution tools payback model



This is clearly a simplistic view but it demonstrates how an initial investment in test execution tools can be of financial benefit in the medium to long term. There are other less tangible benefits as well. However, to get this financial benefit you will need to be sure that there is a requirement to run the same (or very similar) regression tests on many occasions.

Test harnesses

Test harnesses (also known as unit test framework tools) are used primarily by developers to simulate a small section of the environment in which the software will operate. They are usually written in-house by developers to perform component or integration testing for a specific purpose. Test harnesses often use 'mock objects' known as 'stubs' (which stub out the need to have other components or systems by returning predefined values) and 'drivers' (which replace the calling component or system and drive transactions, messages and commands through the software under test).

Test harnesses can be used to test various systems or objects ranging from a middleware system (as in Figure 6.5) to a single or small group of components.

USE IN HOTEL CHAIN SCENARIO

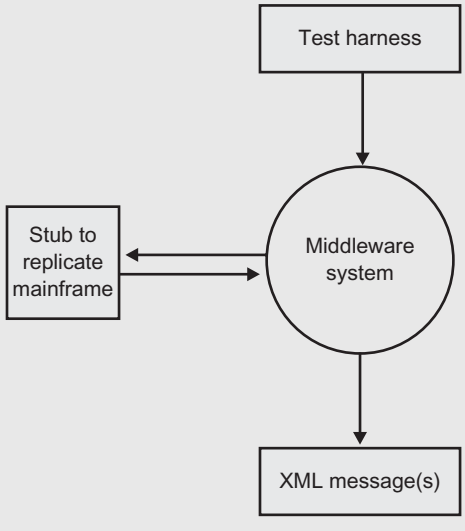
Bookings are entered via the web or GUI front-ends and are loaded onto the mainframe. An overnight batch runs on the mainframe and generates XML messages that are then processed by the middleware system, which makes a further call to the mainframe to read other data. The middleware system then generates further XML messages, which are processed by other systems, resulting in the production of letters to customers.

There are several benefits that can be obtained from using a test harness that generates the XML messages produced by the mainframe:

- It would take a lot of time and effort to design and execute test cases on the mainframe system and run the batch.
- It would be costly to build a full environment.
- The mainframe code needed to generate the XML messages may not yet be available.
- A smaller environment is easier to control and manage. It enables developers (or testers) to perform component and integration testing more quickly as it is easier to localise defects. This allows a quicker turnround time for fixing defects.

The diagram in Figure 6.5 shows that a test harness has been built using a spreadsheet and macros (the driver) to generate XML messages and send them to the middleware. A stub is used to simulate the calls made by the middleware to the mainframe. The contents of the XML messages produced by the middleware can then be compared with the expected results.

Figure 6.5 Test harness for middleware



There are similarities with the principle behind data-driven testing using test execution tools, as the harness allows many different test cases to be designed and run without the time-consuming process of keying them manually. This raises the question of how much benefit can be obtained from using a test execution tool when a test harness can be used instead. As usual, it depends on the circumstances, the risk, the purpose and the level of testing being performed.

Coverage measurement tools

Coverage measurement tools measure the percentage of the code structure covered across white-box measurement techniques such as statement coverage and branch or decision coverage. In addition, they can also be used to measure coverage of modules and function calls. Coverage measurement tools are often integrated with static and dynamic analysis tools and are primarily used by developers.

Coverage measurement tools can measure code written in several programming languages, but not all tools can measure code written in all languages. They are useful for reporting coverage measurement and can therefore be used to assess test completion criteria and/or exit criteria.

Coverage measurement of requirements and test cases/scripts run can usually be obtained from test management tools. This function is unlikely to be provided by coverage measurement tools.

Coverage measurement can be carried out using intrusive or non-intrusive methods. Non-intrusive methods typically involve reviewing code and running code. Intrusive methods, such as 'instrumenting the code' involve adding extra statements into the code. The code is then executed and the extra statements write back to a log in order to identify which statements and branches have been executed.

Instrumentation code can then be removed before it goes into production.

Intrusive methods can affect the accuracy of a test because, for example, slightly more processing will be required to cope with the additional code. This is known as the probe effect and testers need to be aware of its consequences and try to keep its impact to a minimum.

USE IN HOTEL CHAIN SCENARIO

Coverage measurement tools are generally used on high-risk and safety-critical systems and therefore would probably not be used in the Hotel Chain Scenario. However, as an example, assume that the exit criteria for a test phase include the criteria shown in Table 6.4.

Table 6.4 Exit criteria

Function	Module risk level	Branch coverage	Statement coverage
BACS	High	100%	100%
Mailshot	Medium	Not specified	100%
Look-up error message/screen navigation	Low	50%	75%

In this case, coverage measurement tools would be the most appropriate method of assessing whether the exit criteria have been met.

Security tools

Security testing tools are used to test the functions that detect security threats and to evaluate the security characteristics of software. The security testing tool would typically be used to assess the ability of the software under test to:

- handle computer viruses;
- protect data confidentiality and data integrity;
- prevent unauthorised access;
- carry out authentication checks of users;
- remain available under a denial of service attack;
- check non-repudiation attributes of digital signatures.

Security testing tools are mainly used to test e-commerce, e-business and websites. For example, a third-party security application such as a firewall may be integrated into a web-based application.

The skills required to develop and use security tools are very specialised and such tools are generally developed and used on a particular technology platform for a particular purpose. Therefore it maybe worth considering the use of specialist consultancies to perform such testing.

Security tools need to be constantly updated, as there are problems solved and new vulnerabilities discovered all the time – consider the number of Windows XP security releases to see the scale of security problems.

USE IN HOTEL CHAIN SCENARIO

Security tools could be used to test that the firewall surrounding the website can prevent disruption from any type of attack that is specified within the security tool.

In addition, encryption of XML messages to validate bank account details could be tested.

CHECK OF UNDERSTANDING

(1) Why is implementing a comparator likely to be cheaper and quicker than implementing a test execution tool?

(2) Why is the use of test execution tools for record and playback not as effective as it may sound?

(3) Are test execution tools likely to be more useful for testing new systems or testing changes to existing systems? Explain why.

(4) Would both a test execution tool and a test harness be appropriate for acceptance testing?

(5) Name three potential benefits from implementing a test execution tool.

(6) Give three reasons why a test harness is an efficient way of testing components.

(7) Which test execution and logging tools are typically used by developers?

(8) Which test execution and logging tools are likely to be used by specialists in the use of that tool?

Tool support for performance and monitoring

Dynamic analysis tools

Dynamic analysis tools are used to detect the type of defects that are difficult to find during static testing. They are typically used by developers, during component testing and component integration testing, to:

- report on the state of software during its execution;
- monitor the allocation, use and deallocation of memory;
- identify memory leaks;
- detect time dependencies;
- identify unassigned pointers;
- check pointer arithmetic.

They are generally used for safety-critical and other high-risk software where reliability is critical.

Dynamic analysis tools are often integrated with static analysis and coverage measurement tools. For example, a developer may want to perform static analysis on code to localise defects so that they can be removed before component test execution. The integrated tool may allow:

- the code to be analysed statically;
- the code to be instrumented;
- the code to be executed (dynamically).

Dynamic analysis tools are usually language specific, so to test code written in C++ you would need to have a version of a dynamic analysis tool that was specific to C++.

The tool could then:

- report static defects;
- report dynamic defects;
- provide coverage measurement figures;
- report upon the code being (dynamically) executed at various instrumentation points.

USE IN HOTEL CHAIN SCENARIO

The hotel chain would probably not use dynamic analysis tools as the benefits for a normal commercial software system (such as this) are relatively small compared with the investment and ongoing costs of dynamic testing tools. However, if static analysis and coverage measurement tools are used then the additional cost of using dynamic analysis tools may be reduced as they usually come in the same package. Another contributory factor in the decision is that the work done during static analysis and coverage measurement may mean that little additional effort is required to run dynamic tests.

Performance testing/load testing/stress testing tools

Performance testing is very difficult to do accurately and in a repeatable way without using test tools. Therefore performance testing tools have been developed to carry out both load testing and stress testing.

Load testing reports upon the performance of a system under test, under various loads and usage patterns. A load generator (which is a type of test driver) can be used to simulate the load and required usage pattern by creating virtual users that execute predefined scripts across one or more test machines. Alternatively, response times or transaction times can be measured under various levels of usage by running automated repetitive test scripts via the user interface of the

system under test. In both cases output will be written to a log. Reports or graphs can be generated from the contents of the log to monitor the level of performance.

Performance testing tools can also be used for stress testing. In this case, the load on the system under test is increased gradually (ramped up) in order to identify the usage pattern or load at which the system under test fails. For example, if an air traffic control system supports 200 concurrent aircraft in the defined air space, the entry of the 201st or 205th aircraft should not cause the whole system to fail.

Performance testing tools can be used against whole systems but they can also be used during system integration test to test an integrated group of systems, one or more servers, one or more databases or a whole environment.

If the risk analysis finds that the likelihood of performance degradation is low then it is likely that no performance testing will be carried out. For instance, a small enhancement to an existing mainframe system would not necessarily require any formal performance testing. Normal manual testing may be considered sufficient (during which poor performance might be noticed).

There are similarities between performance testing tools and test execution tools in that they both use test scripts and data-driven testing. They can both be left to run unattended overnight and both need a heavy upfront investment, which will take some period of time to pay back.

Performance testing tools can find defects such as:

- General performance problems.
- Performance bottlenecks.
- Memory leakage (e.g. if the system is left running under heavy load for some time).
- Record-locking problems.
- Concurrency problems.
- Excess usage of system resources.
- Exhaustion of disk space.

The cost of some performance tools is high and the implementation and training costs are also high. In addition, finding experts in performance testing is not that easy. Therefore it is worth considering using specialist consultancies to come in and carry out performance testing using such tools.

USE IN HOTEL CHAIN SCENARIO

The likelihood of poor website performance and the cost of lost business and reputation are likely to be sufficient to justify the use of performance testing to mitigate this risk. Performance testing can range from using a relatively cheap

tool to indicate whether performance has improved or deteriorated as a result of the enhancement, to an extensive assessment of response times under normal or maximum predicted usage and identification of the usage pattern that will cause the system to fail.

It is likely that performance test tools will have been used when the website was first developed. Therefore it may be easy to reuse existing tools to do a regression test of performance. If performance tools were not used when the website was developed it is unlikely to be worthwhile buying and implementing expensive performance testing tools.

An alternative option would be to use tools that already exist on servers or in the test environment to monitor performance. It may also be worth considering using external consultants.

Monitoring tools

Monitoring tools are used to check whether whole systems or specific system resources are available and whether their performance is acceptable. Such tools are mainly used in live rather than test environments and are therefore not really testing tools. They tend to be used for monitoring e-commerce, e-business or websites as such systems are more likely to be affected by factors external to the organisation and the consequences can be severe in terms of business lost and bad publicity. Generally, if a website is not available, customers will not report it but will go elsewhere. For example, it was reported in 2003 that a well-known online retailer would lose sales of $660,000 per hour if it were offline during the US trading day.

The use of monitoring tools is generally less important for internal systems as failure is more likely to be noticed only within the organisation and contingency plans may also exist. However, the availability of monitoring tools on mainframes, servers and other forms of hardware, means that it is relatively easy to monitor the majority of an organisation's infrastructure.

USE IN HOTEL CHAIN SCENARIO

A monitoring tool may be beneficial to monitor the website. A monitoring tool may also exist as part of the mainframe system. However, it is less likely that monitoring tools will be used for the GUI front-end that is used by internal staff.

Data quality assessment tools

Data quality assessment tools allow files and databases to be compared against a format that is specified in advance. They are typically used on large scale data intensive projects such as:

- Conversion of data used on one system into a format suitable for another system.

- Migration of data from one system to another.
- Loading of data into a data warehouse.

Data Quality Assessment tools are not specifically designed for testing purposes. They are used primarily for the migration of production data, but typically the development and testing of a migration project will also use these tools.

USE IN HOTEL CHAIN SCENARIO

Suppose that the hotel chain buys a smaller group of hotels, 'Small Hotel Group'.

It could use a data quality assessment tool during the development and testing of an enhancement to its existing systems to include the additional hotels.

The data quality assessment tools could be configured to establish whether the customer data being migrated meets particular quality requirements. These requirements may include:

- valid postcodes;
- valid title for gender;
- numeric values in financial fields;
- numeric values in date fields.

The tool could also be used to reconcile file record counts with data held in header and footer records to confirm that the number of records in the file equals the number of records loaded into the database.

Usability test tools

Usability test tools typically record the mouse clicks made by remote usability testers when carrying out a specified task. Some tools also enable other data to be captured such as screenshots, written comments and voice recordings of verbal comments. This recorded data is generally stored in a database so that it can then be analysed easily by staff at the organisation commissioning the testing.

Note that the usability testing tool market is changing very quickly and new types of usability tools may appear over the next few years.

USE IN HOTEL CHAIN SCENARIO

The changes to the website to improve usability could be tested by a specialist usability testing company who employ, say 50, remote users. The remote users would be given a high-level requirement that would exercise the website changes such as:

- Go to a specified test URL and book three rooms from 3 August to 5 August and two rooms from 7 August. Pay by credit card XYZ.

The mouse clicks, other inputs, and comments recorded by the 50 remote users in carrying out this task would be stored in a database and an analysis report produced by the specialist usability testing company for the hotel chain. This analysis could highlight poor areas of usability in the test website, which could be improved before being deployed to the live website.

CHECK OF UNDERSTANDING

(1) Describe two types of defect that can typically be found using dynamic analysis tools.

(2) Describe two drawbacks associated with performance testing tools.

(3) Which of the tools that provide support for performance and monitoring is most likely to be used by developers?

Other tools

Other tools that are not designed specifically for testers or developers can also be used to support one or more test activities. These include:

- Spreadsheets
- Word processors
- Email
- Back-up and restore utilities
- SQL and other database query tools
- Project planning tools
- Debugging tools (although these are more likely to be used by developers than testers).

For example, in the absence of test management or incident management tools, defects could be recorded on word processors and could be tracked and maintained on spreadsheets. Tests passed or failed could also be recorded on spreadsheets.

USE IN HOTEL CHAIN SCENARIO

Other software tools could also be used:

- A spreadsheet could be used for producing decision tables or working out all the different test scenarios required. It could also be used to manipulate test management information so that it can be presented in the format required in weekly or daily test progress reports.

- Word processors could be used for writing test strategies, test plans, weekly reports and other test deliverables.
- Email could be used for communicating with developers about defects and for distributing test reports and other deliverables.
- Back-up and restore utilities could be used to restore a consistent set of data into the test environment for regression testing.
- SQL could be used for analysing the data held in databases in order to obtain actual or expected results.
- Project planning tools could be used to estimate resources and timescales and monitor progress.
- Debugging tools can be used by developers to localise and fix defects.

CHECK OF UNDERSTANDING

Name four tools that are not specifically designed for testers. Give an example of how each of them could be of use to a tester.

Summary of test tools

Table 6.5 summarises the types of test tools discussed above. It includes the definition given in the ISTQB Glossary of Testing Terms v2.0 and gives a guide to:

- the main ISTQB syllabus classification;
- the activity in the fundamental test process for which the tool is usually most useful;
- the most likely users of the tool.

Table 6.5 Types of test tool

Tool type	ISTQB Syllabus classification	Activity in fundamental test process where it is usually most beneficial	ISTQB Glossary of Testing Terms definition	Most likely users
Test Management tool	Management of testing and tests	All activities	A tool that provides support to the test management and control part of a test process. It often has several capabilities, such as testware management, scheduling of tests, the logging of results, progress tracking, incident management and test reporting.	Testers
Incident management tool	Management of testing and tests	Implementation and execution	A tool that facilitates the recording and status tracking of defects. These tools often have workflow-oriented facilities to track and control the allocation, correction and retesting of defects and provide reporting facilities. Also known as defect-tracking tools.	Various, but particularly testers
Requirements management tool	Management of testing and tests	Analysis and design	A tool that supports the recording of requirements, requirements attributes (e.g. priority, knowledge responsible) and annotation, and facilitates traceability through layers of requirements and requirements change management. Some requirements management tools also provide facilities for static analysis, such as consistency checking and violations to predefined requirements rules.	Various, but particularly business analysts

(Continued)

Table 6.5 *(Continued)*

Tool type	ISTQB Syllabus classification	Activity in fundamental test process where it is usually most beneficial	ISTQB Glossary of Testing Terms definition	Most likely users
Configuration management tool	Management of testing and tests	Implementation and execution	Not defined.	Various
Review tool	Static testing	Implementation and execution	A tool that provides support to the review process. Typical features include review planning and tracking support, communication support, collaborative reviews and a repository for collecting and reporting of metrics.	Various
Static analysis tools	Static testing	Implementation and execution	Performs and supports analysis of software artifacts, e.g. requirements or code, carried out without execution of these software artifacts.	Developers
Modelling tools	Static testing	Implementation and execution	Not defined.	Developers
Test design tools/script generators	Test specification	Analysis and design	A tool that supports the test design activity by generating test inputs from a specification that may be held in a CASE tool repository, e.g. requirements management tool, or from specified test conditions held in the tool itself.	Testers

(Continued)

Table 6.5 *(Continued)*

Tool type	ISTQB Syllabus classification	Activity in fundamental test process where it is usually most beneficial	ISTQB Glossary of Testing Terms definition	Most likely users
Test oracles (considered to be a subset of test design tools)	Test specification	A nalysis and design	A source to determine expected results to compare with the actual result of the software under test. An oracle may be the existing system (for a benchmark), a user manual, or an individual's specialised knowledge, but should not be the code.	Testers
Test (input) data preparation tools	Test specification	Analysis and design	A type of test tool that enables data to be selected from existing databases or created, generated, manipulated and edited for use in testing.	Various
Test execution/ test running tools	Test execution and logging	Implementation and execution	A type of test tool that is able to execute other software using an automated test script, e.g. capture/playback.	Testers
Test harness/ unit test framework tools (stubs and drivers)	Test execution and logging	Implementation and execution	A test environment composed of stubs and drivers needed to conduct a test.	Developers
Test comparators	Test execution and logging	Implementation and execution	A test tool used to support and/or automate the process of identifying differences between the actual results produced by the component or system under test and the expected results for a test. Test comparison can be performed during test execution (dynamic comparison) or after test execution.	Testers and developers

(Continued)

Table 6.5 *(Continued)*

Tool type	ISTQB Syllabus classification	Activity in fundamental test process where it is usually most beneficial	ISTQB Glossary of Testing Terms definition	Most likely users
Coverage measurement tools	Test execution and logging	Implementation and execution	A tool that provides objective measures of what structural elements, e.g. statements, branches, have been exercised by a test suite.	Developers
Security tools	Test execution and logging	Implementation and execution	A tool that supports testing to determine the security of the software product. Security is defined as: attributes of software products that bear on its ability to prevent unauthorised access, whether accidental or deliberate, to programs and data.	Security testing specialists
Dynamic analysis tools	Performance and monitoring	Implementation and execution	A tool that provides run-time information on the state of the software code. These tools are most commonly used to identify unassigned pointers, check pointer arithmetic and to monitor the allocation, use and deallocation of memory and to flag memory leaks.	Developers
Performance testing/load testing/stress testing tools	Performance and monitoring	Implementation and execution	A tool to support performance testing and that usually has two main facilities: load generation and test transaction measurement. Load generation can simulate either multiple users or high volumes of input data. During execution, response time measurements are taken from selected transactions and these are logged. Performance testing tools normally provide reports based on test logs and graphs of load against response times.	Performance testing specialists

(Continued)

Table 6.5 *(Continued)*

Tool type	ISTQB Syllabus classification	Activity in fundamental test process where it is usually most beneficial	ISTQB Glossary of Testing Terms definition	Most likely users
Monitoring tools	Performance and monitoring	Implementation and execution	A software tool or hardware device that runs concurrently with the component or system under test and supervises, records and/or analyses the behaviour of the component or system.	Various
Data Quality Assessment tools	Data quality	Implementation and execution	No defined.	Various
Usability tools	Usability	Implementation and execution	Not defined. (Usability Testing is defined as testing to determine the extent to which the software product is understood, easy to learn, easy to operate and attractive to the users under specified conditions.)	Usability Testing specialists
Spreadsheets	Other tools	All activities	Not defined.	Various
SQL	Other tools	Implementation and execution	Not defined.	Various
Project planning/ resource	Other tools	Planning and control	Not defined.	Various
Debugging tools	Other tools	Not used for testing activities	A tool used by programmers to reproduce failures, investigate the state of programs and find the corresponding defect. Debuggers enable programmers to execute programs step by step, to halt a program at any program statement and to set and examine program variables.	Developers

INTRODUCING A TOOL INTO AN ORGANISATION

There are many stages in the process that should be considered before implementing a test tool.

Analyse the problem/opportunity

An assessment should be made of the maturity of the test process used within the organisation. If the organisation's test processes are immature and ineffective then the most that the tool can do is to make the repetition of these processes quicker and more accurate – quick and accurate ineffective processes are still ineffective!

It is therefore important to identify the strengths, weaknesses and opportunities that exist within the test organisation before introducing test tools. Tools should only be implemented that will either support an established test process or support required improvements to an immature test process. It may be beneficial to carry out some TPI (Test Process Improvement) or CMMi (Capability Maturity Model Integration) assessment to establish the maturity of the organisation before considering the implementation of any test tool.

Generate alternative solutions

It may be more appropriate and cost-effective to do something different. In some organisations, performance testing, which may only need to be done from time to time, could be outsourced to a specialist testing consultancy. Training or recruiting better staff could provide more benefits than implementing a test tool and improve the effectiveness of a test process more significantly. In addition, it is more effective to maintain a manual regression pack so that it accurately reflects the high-risk areas than to automate an outdated regression pack (that is no longer relevant) using a test execution tool.

An early investigation of what tools are available is likely to form part of this activity.

Constraints and requirements

A thorough analysis of the constraints and requirements of the tool should be performed. Interested parties should attend workshops and/or be interviewed so that a formal description of the requirements can be produced and approved by the budget holder and other key stakeholders.

A failure to specify accurate requirements (as with a failure to specify accurate requirements for a piece of software) can lead to delays, additional costs and the wrong things being delivered. This could lead to a review tool being implemented that does not allow access across the internet, even though there is a need for staff from many countries to participate in reviews. Any financial or technical constraints (e.g. compatibility with particular operating systems or databases) should also be considered.

It is useful to attach some sort of priority or ranking to each requirement or group of requirements.

Training, coaching and mentoring requirements should also be identified. For example, experienced consultants could be used for a few weeks or months to work on overcoming implementation problems with the tool and to help transfer knowledge to permanent staff. Such consultants could be provided by the vendor or could be from the contract market.

Requirements for the tool vendor should also be considered. These could include the quality of training and support offered by the vendor during and after implementation and the ability to enhance and upgrade the tool in the future. In addition, their financial stability should be considered as the vendor could go bankrupt or sell to another vendor. Therefore, using a small niche vendor may be a higher risk than using an established tool supplier.

If non-commercial tools (such as open source and freeware) are being considered then there are likely to be risks around the lack of training and support available. In addition, the ability or desire of the service support supplier (or open-source provider) to continue to develop and support the tool should be taken into account.

Evaluation and shortlist

The tools available in the marketplace should be evaluated to identify a shortlist of the tools that provide the best fit to the requirements and constraints. This may involve:

- searching the internet;
- attending exhibitions of test tools;
- discussions with tool vendors;
- engaging specialist consultants to identify relevant tools.

It may also be useful for the test organisation to send a copy of its list of requirements and constraints to tool vendors so that:

- the vendor is clear about what the test organisations wants;
- the vendor can respond with clarity about what its own tools can do and what workarounds there are to meet the requirements that the tool cannot provide;
- the test organisation does not waste time dealing with vendors that cannot satisfy its key requirements.

The outcome of this initial evaluation should result in a shortlist of perhaps one, two or three tools that appear to meet the requirements.

Detailed evaluation/proof of concept

A more detailed evaluation (proof of concept) should then be performed against this shortlist. This should be held at the test organisation's premises in the test environment in which the tool will be used. This test environment should use the system under test and other software, operating systems and hardware with which the tool will be used. There are several reasons why there is little benefit from evaluating the tool on something different. For example:

- Test execution tools do not necessarily recognise all object types in the system under test, or they may need to be reconfigured to do so.
- Performance measurement tools may need to be reconfigured to provide meaningful performance information.
- Test management tools may need to have workflow redesigned to support established test processes and may need to be integrated with existing tools used within the test process.
- Static analysis tools may not work on the version of programming languages used.

In some cases, it may be worth considering whether changes can be made to the organisation's test environments and infrastructure, but the costs and risks need to be understood and quantified.

(Note that if there is only one tool in the shortlist then it may be appropriate to combine the proof of concept and the pilot project.)

After each proof of concept the performance of the tool should be assessed in relation to each predefined requirement. Any additional features demonstrated should be considered and noted as potential future requirements.

Once all proofs of concept have been carried out it may be necessary to amend the requirements as a result of what was found during the tool selection process. Any amendments should be agreed with stakeholders. Each tool should then be assessed against the finalised set of requirements.

There are three likely outcomes at this stage:

- None of the tools meet the requirements sufficiently well to make it worthwhile purchasing and implementing them.
- One tool meets the requirement much better than the others and is likely to be worthwhile. In this case select this tool.
- The situation is unclear and more information is needed. In this case a competitive trial or another cycle/iteration of the process may be needed. Perhaps the requirements need to be revised or further questions need to be put to vendors. It may also be time to start negotiations with vendors about costs.

Negotiations with vendor of selected tool

Once a tool has been selected discussions will be held with the vendor to establish and negotiate the amount of money to be paid and the timing of payments. This will include some or all of the following:

- purchase price;
- annual licence fee;
- consultancy costs;
- training costs;
- implementation costs.

Discussions should establish the amount to be paid, first, for a pilot project and, secondly (assuming the pilot project is successful), the price to be paid for a larger scale implementation.

The pilot project
The aims of a pilot project include the following:

- It is important to establish what changes need to be made to the high-level processes and practices currently used within the test organisation. This involves assessing whether the tool's standard workflow, processes and configuration need to be amended to fit with the test process or whether the existing processes need to be changed to obtain the optimum benefits that the tool can provide.
- To determine lower level detail such as templates, naming standards and other guidelines for using the tool. This can take the form of a user guidelines document.
- To establish whether the tool provides value for money. This is done by trying to estimate and quantify the financial and other benefits of using the tool and then comparing this with the fees paid to the vendor and the projected internal costs to the organisation (e.g. lost time that could be used for other things, the cost of hiring contractors, etc.).
- A more intangible aim is to learn more about what the tool can and cannot do and how these functions (or workarounds) can be applied within the test organisation to obtain maximum benefit.

The pilot project should report back to the group of stakeholders that determined the requirements of the tool.

If a decision is made to implement the tool on a larger scale then a formal project should be created and managed according to established project management principles. (This is outside the scope of the book and the syllabus. See ISEB Project Management.)

Key factors in successful implementations of test tools
There are certain factors or characteristics that many successful tool implementation projects have in common:

- Implementing findings from the pilot project such as high-level process changes and using functions or workarounds that can add additional benefits.
- Identifying and subsequently writing user guidelines, based on the findings of the pilot project.
- An incremental approach to rolling out the tool into areas where it is likely to be most useful. For example, this can allow 'quick wins' to be made and good publicity obtained, resulting in a generally positive attitude towards the tool.
- Improving the process to fit with the new tool, or amending the use of the tool to fit with existing processes.

- Ensuring that the appropriate level of training, coaching and mentoring is available. Similarly, there may be a need to recruit permanent or contract resources to ensure that sufficient skills exist at the outset of the tool's use within the organisation.
- Using a database (in whatever format) of problems encountered and lessons learnt to overcome them. This is because new users are likely to make similar mistakes.
- Capturing metrics to monitor the amount of use of the tool. Recording the benefits obtained. This can then be used to support arguments about implementing to other areas within the test organisation.
- Agreeing or obtaining a budget to allow the tool to be implemented appropriately.

Summary of test tool implementation process

The diagram in Figure 6.6 outlines the process for selecting and implementing a test tool in an organisation. This shows that there are several points at which a decision could be made **not** to introduce a tool. It also demonstrates that the activities during the evaluation and negotiation stages can follow an iterative process until a decision is made.

CHECK OF UNDERSTANDING

(1) Why is an understanding of the test organisation's maturity essential before introducing a test tool?

(2) What is the purpose of defining requirements for the tool?

(3) Why is it important to evaluate the tool vendor as well as the tool itself?

(4) What is meant by a proof of concept?

(5) What is the purpose of a pilot project?

(6) When is it appropriate to combine a proof of concept and pilot project?

(7) Name three factors in the successful implementation of tools.

SUMMARY

We have seen that the main benefits of using test tools are generally the same as the benefits from automating a process in any industry. These are: time saved; and predictable and consistent results.

However, we have also seen that there can be considerable costs in terms of both time and money associated with obtaining such benefits. The point at which the use of tools becomes economically viable depends on the amount of reuse, which is often difficult to predict.